Some New Tractable Classes of CSPs and their Relations with Backtracking Algorithms

Achref El Mouelhi, Philippe Jegou, Cyril Terrioux, Bruno Zanuttini

Some New Tractable Classes of CSPs and their Relations with Backtracking Algorithms *

Achref El Mouelhi[1], Philippe Jégou[1], Cyril Terrioux[1], and Bruno Zanuttini[2]

[1] LSIS - UMR CNRS 7296
Aix-Marseille Université
Avenue Escadrille Normandie-Niemen
13397 Marseille Cedex 20 (France)
{achref.elmouelhi, philippe.jegou, cyril.terrioux}@lsis.org
[2] Normandie Université, France
GREYC, Université de Caen Basse-Normandie, CNRS UMR 6072, ENSICAEN
Campus II, Boulevard du Maréchal Juin
14032 Caen Cedex (France)
bruno.zanuttini@unicaen.fr

Abstract. In this paper, we investigate the complexity of algorithms for solving CSPs which are classically implemented in real practical solvers, such as Forward Checking or Bactracking with Arc Consistency (RFL or MAC).. We introduce a new parameter for measuring their complexity and then we derive new complexity bounds. By relating the complexity of CSP algorithms to graph-theoretical parameters, our analysis allows us to define new tractable classes, which can be solved directly by the usual CSP algorithms in polynomial time, and without the need to recognize the classes in advance. So, our approach allows us to propose new tractable classes of CSPs that are naturally exploited by solvers, which indicates new ways to explain in some cases the practical efficiency of classical search algorithms.

1 Introduction

Constraint Satisfaction Problems (CSPs [1]) constitute an important formalism of Artificial Intelligence (AI) for expressing and efficiently solving a wide range of practical problems. A constraint network (or CSP, abusing words) consists of a set of variables X, each of which must be assigned a value in its associated (finite) domain D, so that these assignments together satisfy a finite set C of constraints.

Deciding whether a given CSP has a solution is an NP-complete problem. Hence classical approaches to this problem are based on backtracking algorithms, whose worst-case time complexity is at best of the order $O(e.d^n)$ with n the number of variables, e the number of constraints and d the size of the largest domain. To increase efficiency, such algorithms also rely on filtering techniques during search (among other techniques, such as variable ordering heuristics). With the

* This work was supported by the French National Research Agency under grant TUPLES (ANR-2010-BLAN-0210).

help of such techniques, despite their theoretical time complexity, algorithms such as Forward Checking [2] (denoted FC), RFL (for Real Full Look-ahead [3]) or MAC (for Maintaining Arc Consistency [4]) for binary CSPs, or nFC$_i$ for non-binary CSPs [5] turn out to be very efficient in practice on a wide range of practical problems.

In a somewhat orthogonal direction, other works have addressed the effectiveness of solving CSPs by defining *tractable classes*. A tractable class is a class of CSPs which can be recognized, and then solved, using polynomial time algorithms. Different kinds of tractable classes have been introduced. Some of them are based on the *structure* of the constraint network, for instance tree-structured networks [6] or more generally, networks of bounded width [7]. This kind of tractable classes has shown its practical interest for benchmarks of small width (e.g. [8]). Other studies have highlighted the interest of certain tractable classes (e.g. [9]) but unfortunately they are very rare. This direction of research has produced and still produces works both numerous and complex and the results are generally difficult to establish (we can find a synthesis in [1]). Unfortunately, most of these results remain only theoretical and thus, the question of their real interest should be raised in the context of constraint programming.

It is not easy to cite one tractable class in the field of CSP that has shown any interest in practice (with the exception of bounded width) and ideally allow it to outperform the efficiency of current solvers. So, we think that it seems necessary today to ask that question to the CPAIOR community, even if this question could be controversial.

In our opinion, the reasons for lack of practical interest of the tractable classes exhibited to date are based on several aspects. Firstly, the identification of a new tractable class requires the development of ad hoc polytime algorithms: one for the recognition of tractable instances, and one for solving them. Secondly, these polytime algorithms are generally neither efficient in practice, and frequently, nor in theory. And most importantly, the proposed tractable classes seem to be artificial in the sense that they do not exist in reality: real benchmarks do not belong to these classes, and even tractable classes only appear in small pieces of real problems, this making them finally completely unusable. And surprisingly, most tractable classes currently exhibited by the community seem to conceal their only interest by their theoretical difficulty. Finally, it seems that these classes have no interest, from a practical point of view, for the CPAIOR community. In addition, to be efficient, solvers need to rely on simple mechanisms that can be efficiently implemented. So, to integrate the use of tractable classes whose treatment would not be in linear time seems almost useless, because their treatment would significantly slow down the efficiency of a solver and thus make it inoperative in practice.

We do not criticize the existence of works on tractable classes, but essentially the direction they take, and we propose to redirect the works in the direction which seems, after several decades of works on the issue, the only one which can be of interest to the CPAIOR community, or at least, offers the best chance of producing useful results. So, we propose here to focus research on the analysis of algorithms such as FC, RFL or nFC$_i$, whose theoretical complexity is exponential, but which are the basis of practical systems for constraint solving,

and whose concrete results are often impressive in terms of computational time. Their analysis could lead to identify tractable classes which could then be efficiently exploited in practice, possibly with some slight modifications of solvers. In this respect, our study is very close in spirit to the study by Rauzy about satisfiability and the behaviour of DPLL on known tractable instances [10], and more recently, of the works presented in [11] for CSPs of bounded structural parameters or in [12] for β-acyclic CNFs in SAT.

We do so by reevaluating their time complexity using a new parameter, namely the number of maximal cliques in the *microstructure* [13] of the instance or in the *generalized microstructure* for the non-binary case. For the binary case, writing $\omega_{\#}(\mu(P))$ for the number of maximal cliques in the microstructure of a CSP P, we show that the complexity of an algorithm such as FC is in $O(n^2 d \cdot \omega_{\#}(\mu(P)))$. This provides a new perspective on the study of the efficiency of backtracking-like algorithms, by linking it to a well-known graph-theoretical parameter. In particular, reusing known results from graph theory, we propose some tractable classes of CSPs. The salient feature of these classes is that they are solved in polynomial time by *general-purpose*, *widely used* algorithms, *without the need for the algorithms to recognize the class*.

The paper is organized as follows. We first introduce notations and recall the definitions and basic properties of the microstructure. Then we present our complexity analysis of BT, FC and RFL on binary CSPs, and we introduce the notion of *generalized microstructure* to extend our study to non-binary CSPs and to algorithms of the class nFC_i. We then point at new tractable classes issued from graph theory, which can be exploited in the field of CSPs. Finally, we give a discussion and perspectives for future work.

2 Preliminaries

Before reviewing the classical analysis of algorithms, we recall some basic notions about CSPs and their microstructure.

Definition 1 (CSP) *A finite* constraint satisfaction problem *(CSP) is a triple* (X, D, C)*, where* $X = \{x_1, \ldots, x_n\}$ *is a set of variables,* $D = (D(x_1), \ldots, D(x_n))$ *is a list of finite domains of values, one per variable, and* $C = \{c_1, \ldots, c_e\}$ *is a finite set of constraints. Each constraint* c_i *is a couple* $(S(c_i), R(c_i))$*, where* $S(c_i) = \{x_{i_1}, \ldots, x_{i_k}\} \subseteq X$ *is the* scope *of* c_i*, and* $R(c_i) \subseteq D(x_{i_1}) \times \cdots \times D(x_{i_k})$ *is its* relation*. The* arity *of* c_i *is* $|S(c_i)|$.

We will refer to a binary constraint with scope $\{x_i, x_j\}$ by the notation c_{ij}. A *binary CSP* is one in which all constraints are binary. Otherwise (general case), the CSP is said to be *n-ary*. We assume that all variables appear at least in one scope of constraint and that for a given scope, there is at most one constraint. This is without loss of generality since two constraints over the same set of variables can be merged into one by taking the intersection of their relations (for the purpose of analysis).

Definition 2 (assignment, solution) *Given a CSP* (X, D, C)*, an* assignment *of values to* $Y \subseteq X$ *is a set of pairs* $t = \{(x_i, v_i) \mid x_i \in Y\}$ *(written* $t =$

$(v_1, \ldots, v_k)$ *when no confusion can arise), with* $v_i \in D(x_i)$ *for all* i*. An assignment to* $Y \subseteq X$ *is said to be* consistent *(or a* partial solution*) if all constraints* $c \in C$ *with scope* $S(c) \subseteq Y$ *are satisfied, i.e.,* $t[S(c)] \in R(c)$ *holds with* $t[S(c)]$ *the restriction of* t *to* $S(c)$*. A* solution *is a consistent assignment to* X*.*

We consistently write n for the number of variables in a CSP, d for the cardinality of the largest domain, e for the number of constraints, a for the maximum arity over all constraints, and r for the number of tuples of the largest relation.

Given a CSP, the basic question is to decide whether it has a solution, which is well-known to be NP-complete. In order to study CSPs and try to circumvent this difficulty, various points of view can be adopted. As concerns *binary* CSPs, one of them is the *microstructure* of an instance, that is its compatibility graph as we define now. Intuitively, the vertices of this graph code the values, and its edges code their compatibility.

Definition 3 (microstructure) *Given a binary CSP* $P = (X, D, C)$*, the* microstructure *of* P *is the undirected graph* $\mu(P) = (V, E)$ *with:*

- $V = \{(x_i, v_i) : x_i \in X, v_i \in D(x_i)\}$,
- $E = \{\ \{(x_i, v_i), (x_j, v_j)\}\ \mid\ i \neq j, c_{ij} \notin C$ *or* $(v_i, v_j) \in R(c_{ij})\}$

In words, the microstructure of a binary CSP P contains an edge for all pairs of vertices, except for vertices coming from the same domain and for vertices corresponding to pairs which are forbidden by some constraint. It can easily be seen that the microstructure of a CSP is an n-partite graph, since there is no edge connecting vertices issued from the same domain. In this paper, we will study the complexity of CSP algorithms through cliques in the microstructure.

Definition 4 (clique) *A* complete graph *is a simple graph in which every pair of distinct vertices is connected by an edge. A* k-clique *in an undirected graph is a subset of* k *vertices inducing a complete subgraph (all the vertices are pairwise adjacent). A* maximal clique *is a clique which is not a proper subset of another clique. We write* $\omega_{\#}(G)$ *for the number of maximal cliques in a graph* G*.*

The following result follows directly from the fact that in a microstructure, the vertices of a clique correspond to compatible values which are by construction issued from different domains.

Proposition 1 *Given a binary CSP* P *and its microstructure* $\mu(P)$*, an assignment* $(v_1, ..., v_n)$ *to* X *is a solution of* P *iff* $\{(x_1, v_1), ..., (x_n, v_n)\}$ *is an* n*-clique of* $\mu(P)$*.*

It can be seen that the transformation of a CSP P to its microstructure $\mu(P)$ can be realized in polynomial time. A polynomial reduction directly follows, from the problem of deciding whether a given CSP has a solution, to the problem of deciding whether a given undirected graph has a clique of a given size (the famous "clique problem"). This transformation has first been exploited by [13], who proposed tractable classes of CSPs based on known tractable classes for the maximal clique problem (*chordal graphs* [14]). A similar approach has been taken for *hybrid* tractable classes [15, 16].

Backtracking algorithms We now briefly review the complexity of algorithms of interest here: BT, FC, RFL and MAC for both binary and non-binary CSPs. These algorithms essentially cover all the approaches which use backtracking and lookahead (variable and value ordering left apart).

The *Backtracking* algorithm (BT, a.k.a. *Chronological Backtracking*) is a recursive enumeration procedure. It starts with an empty assignment and in the general case, given a current partial solution $(v_1, v_2, \ldots, v_i)$, it chooses a new variable x_{i+1} and tries to assign values of $D(x_{i+1})$ to x_{i+1}. The only check performed while doing so is that the resulting assignment $(v_1, v_2, \ldots, v_i, v_{i+1})$ is consistent. In the affirmative, BT continues with this new partial solution to a new unassigned variable (called a *future variable*). Otherwise (if $(v_1, v_2, \ldots, v_i, v_{i+1})$ is not consistent), BT tries another value from $D(x_{i+1})$. If there is no such unexplored value, BT is in a dead-end, and then it uninstantiates x_i (it performs a *backtrack*). It is easily seen that the search performed by BT corresponds to a depth-first traversal of a semantic tree called the *search tree*, whose root is an empty tuple, while the nodes at the i^{th} level are i-tuples which represent the assignments of the variables along the corresponding path in the tree. Nodes in this tree which correspond to partial solutions are called *consistent nodes*, while other nodes are called *inconsistent nodes*. The number of nodes in the search tree is at most $\Sigma_{0 \leq i \leq n} d^i = \frac{d^{n+1}-1}{d-1}$, hence it is in $O(d^n)$. So, the complexity of BT can be bounded by the number of nodes multiplied by the cost at each node. Assuming that a constraint check can be achieved in $O(a)$, the complexity of BT is $O(e.a.d^n)$.

BT can be considered as a generic algorithm. Algorithms based on BT and used in practice perform some extra work at each node in the search tree, namely, they remove inconsistent values from the domain of future variables (filtering). For *binary* CSPs, FC removes values inconsistent with the current assignment, and RFL moreover enforces full arc-consistency (AC) on the future variables. The complexity of FC can be bounded by $O(nd^n)$. Using an $O(ed^2)$ algorithm for achieving AC, the complexity of RFL is in $O(ed^2d^{n-1}) = O(ed^{n+1})$. For *n-ary* CSPs, the algorithms of the class $\text{nFC}_i (i = 0, 1 \ldots 5)$ cover the partial and total enforcement of *generalized* arc consistency (GAC) on a subset of constraints involving both assigned variables and future variables. In each case, the filtering is achieved after each variable assignment. So, the complexity of nFC_i depends on the cost of the filtering. Hence, for nFC_5 which achieves the most powerful filtering, the complexity is in $O(ead^n)$. It is the same if we consider the non-binary version of RFL which maintains GAC at each node. In the following, we denote by nBT and nRFL the non-binary versions of BT and RFL.

The algorithm M(G)AC (for Maintaining (Generalized) Arc-Consistency) is slightly different from previous algorithms. Assume that an assignment (x_{i+1}, v_{i+1}) (called a *positive decision*) produces a dead-end. After returning to the current assignment $(v_1, v_2, \ldots v_i)$, and before assigning a new value to x_{i+1}, the value v_{i+1} is deleted from the domain $D(x_{i+1})$, and a (G)AC filtering is realized. All the domains of future variables can be impacted by this filtering. This process is called *refutation* of value x_{i+1}, and can be understood as extending the current partial solution $(v_1, v_2, \ldots, v_i)$ with the "negative" child $(x_{i+1}, \neg v_{i+1})$ (called a

negative decision), then enforcing arc-consistency, which can end up in a dead end or in further exploration.

Hence the structure and the size of a MAC search tree are different from previous algorithms. First, it is a binary tree verifying particular properties. Each branch of the search tree corresponds to a set of decisions $\Delta = \{\delta_1, \ldots, \delta_i\}$ where each δ_j may be a positive or negative decision. Given an internal node, a negative decision $(x_{i+1}, \neg v_{i+1})$ is produced only after a dead-end has occurred with the positive decision (x_{i+1}, v_{i+1}). Thus, the number of nodes issued from a negative decision is at most the number of nodes issued from standard assignments. Hence for MAC, the number of nodes in the search tree is at most $2 \times \Sigma_{0 \leq i \leq n-1} d^i = 2\frac{d^n-1}{d-1} \in O(d^{n-1})$. Since the cost associated to a node is bounded by the cost of the AC filtering $O(ed^2)$, the worst-case complexity of MAC is the same as for RFL, that is $O(ed^{n+1})$.

We note here that algorithms (n)BT, FC, nFC$_i$, (n)RFL, or M(G)AC may use a *dynamic* variable ordering, that is, which variable (x_{i+1}) to explore next can typically be decided on each assignment.

3 New Complexity Analysis for Binary CSPs

We now come to the heart of our contribution, namely, a complexity analysis of classical algorithms in terms of parameters related to the microstructure. In the following, we say that a node of the search tree is a *maximally deep consistent node* if it is consistent and has no consistent child node (on the next variable in the ordering). Hence, such a node corresponds either to a solution, or to a partial solution which cannot be consistently extended to the next variable. The following result is central to our study.

Proposition 2 *Given a binary CSP $P = (X, D, C)$, there is an injective mapping from the maximally deep consistent nodes explored by BT onto the maximal cliques in $\mu(P)$.*

Proof: Let $(v_1, v_2, \ldots, v_i)$ be a maximally deep consistent node explored by BT. By definition, $(v_1, v_2, \ldots, v_i)$ is a partial solution, hence for all $1 \leq j, k \leq i$, either there is no constraint c_{jk} in C with scope $\{x_j, x_k\}$, or (v_j, v_k) is in the relation $R(c_{jk})$. In both cases $\{(x_j, v_j), (x_k, v_k)\}$ is an edge in $\mu(P)$. Hence $\{(x_1, v_1), \ldots, (x_i, v_i)\}$ is a clique in $\mu(P)$ and hence, is included in some maximal clique of $\mu(P)$. Write $Cl(v_1, v_2, \ldots, v_i)$ for an arbitrary one.

We now show that Cl forms an injective mapping from maximally deep consistent nodes to maximal cliques. By construction of BT, if $(v_1, v_2, \ldots, v_i)$ and $(v'_1, v'_2, \ldots, v'_{i'})$ are two maximally deep nodes explored, then they must differ on the value of at least one variable. Precisely, they must differ at least at the point where the corresponding paths split in the search tree, corresponding to some variable x_j assigned to some value on one path, and to some other value on the other one[3]. Since there are no edges in $\mu(P)$ connecting two values of the

[3] We use at this point the assumption that the algorithms explore all the values of a variable before reordering the future variables.

same variable, there cannot be a maximal clique containing both $(v_1, v_2, \ldots, v_i)$ and $(v'_1, v'_2, \ldots, v'_{i'})$, hence Cl is injective. □

Using this proposition, we can easily bound the number of nodes in a search tree induced by a backtracking search, and its time complexity, in terms of the microstructure. As is common, we assume that a constraint check (deciding $(v_i, v_j) \in R(c_{ij})$) requires constant time.

Proposition 3 *The number of nodes $N_{BT}(P)$ in the search tree developed by BT for solving a given binary CSP $P = (X, D, C)$, satisfies $N_{BT}(P) \leq nd \cdot \omega_{\#}(\mu(P))$. Its time complexity is in $O(n^2 d \cdot \omega_{\#}(\mu(P)))$.*

Proof: First consider the number of consistent nodes. Because any node in the search tree is at depth at most n and the path from the root to a consistent node contains only consistent nodes, as a direct corollary of Proposition 2 we obtain that the search tree contains at most $n \cdot \omega_{\#}(\mu(P))$ consistent nodes. Now by definition of BT, a consistent node has at most d children (one per candidate value for the next variable), and inconsistent nodes have none. It follows that the search tree has at most $nd \cdot \omega_{\#}(\mu(P))$ nodes of any kind.

The time complexity follows directly, since each node corresponds to extending the current partial assignment to one more variable (x_{i+1}), which involves at most one constraint check per other variable (check $c_{j(i+1)}$ for each x_j already assigned).□

It can be seen that in the statement of Proposition 3 and in the forthcoming ones, the number of maximal cliques $\omega_{\#}(\mu(P))$ could be replaced by the number of maximal cliques *of size at most* $n - 1$. This is because as soon as a path is explored which is contained in an n-clique, that is, in a solution, no backtracking will occur further than this path.

We now turn to forward checking and RFL. Clearly enough, Proposition 2 also holds for both. The number of nodes explored follows from the fact that only consistent nodes are explored. The time complexity follows from the fact that at most n future domains are filtered by FC, and AC is enforced in time $O(ed^2)$ by RFL.

Proposition 4 *The number of nodes $N_{FC}(P)$ in the search tree developed by FC or by RFL for solving a given CSP $P = (X, D, C)$, satisfies $N_{FC}(P) \leq n \cdot \omega_{\#}(\mu(P))$. The time complexity of FC is in $O(n^2 d \cdot \omega_{\#}(\mu(P)))$, and that of RFL is in $O(ned^2 \cdot \omega_{\#}(\mu(P)))$.*

Regarding MAC, unfortunately, the existence of negative decisions in the tree search makes that the proof is not so easy as for other algorithms. So, at present time, we can only claim a conjecture about this result. As a consequence, we assume too that the time complexity could be a function linear in the number of maximal cliques in the microstructure.

Conjecture 1 *Given a binary CSP $P = (X, D, C)$, there is an injective mapping from the maximally deep consistent nodes explored by MAC onto the maximal cliques in $\mu(P)$.*

4 New Analysis for Non-Binary CSPs

We now turn to n-ary CSPs. We first extend the notion of microstructure to this general case, then we analyze the complexity of nFC$_i$ in terms of maximal number of cliques.

4.1 A Generalized Microstructure

Note that a generalization of the notion of microstructure was first proposed in [15]. Nevertheless, this notion is based on hypergraphs and has been little used so far. In contrast, our notion sticks to the simpler framework of graphs. Our *generalized microstructure* is essentially obtained by letting vertices encode the tuples (from relations involved in the CSP) rather than unary assignments (x_i, v_i) of the binary case. Note that other generalizations have been proposed (see [17]) but due to lack of space, we cannot treat them here.

Definition 5 (generalized microstructure) *Given a CSP $P = (X, D, C)$ (not necessarily binary), the* generalized microstructure *of P is the undirected graph $\mu_G(P) = (V, E)$ with:*

- $V = \{(c_i, t_i) : c_i \in C, t_i \in R(c_i)\}$,
- $E = \{\ \{(c_i, t_i), (c_j, t_j)\}\ \mid\ i \neq j, t_i[S(c_i) \cap S(c_j)] = t_j[S(c_i) \cap S(c_j)]\}$

Like for the microstructure, there is a direct relationship between cliques and solutions of CSPs;

Proposition 5 *A CSP P has a solution iff $\mu_G(P)$ has a clique of size e.*

Proof: By construction, $\mu_G(P)$ is e-partite, and any clique contains at most one vertex (c_i, t_i) per constraint $c_i \in C$. Hence the e-cliques of $\mu_G(P)$ correspond exactly to its cliques with one vertex (c_i, t_i) per constraint $c_i \in C$. Now by construction of $\mu_G(P)$ again, any two vertices (c_i, t_i), (c_j, t_j) joined by an edge (in particular, in some clique) satisfy $t_i[S(c_i) \cap S(c_j)] = t_j[S(c_i) \cap S(c_j)]$. Hence all t_i's in a clique join together, and it follows that the e-cliques of $\mu_G(P)$ correspond exactly to tuples t which are joins of one allowed tuple per constraint, that is, to solutions of P. □

One can observe that the generalized microstructure corresponds to the microstructure of the dual representation of a CSP [18]. One can also see that our generalization is in fact the *line-graph* (the dual graph) of the hypergraph proposed in [15], in which we add edges for pairs of constraints whose scopes have empty intersections. Hence, in the same spirit, we can propose other generalizations of the microstructure by considering any graph-based representation of a non-binary CSP as soon as the representation has the same set of solutions - wrt. a given bijection - as the original instance (e.g. the hidden variable encoding [19]). However, due to lack of space, we do not deal with these issues here.

4.2 Time Complexity of nBT, nFC and nRFL

We now investigate the complexity of algorithms for solving n-ary CSPs. We first make an assumption about the order in which such algorithms explore variables, then we discuss this restriction.

Definition 6 (compatible with constraints) *Let P be a CSP. A total order $(x_1, x_2, \dots x_n)$ on X is said to be* compatible with the constraints in C *if there are k constraints $c_{i_1}, c_{i_2}, \dots c_{i_k}$ in C ($1 \leq k \leq e$) which satisfy:*

- $\bigcup_{1 \leq \ell \leq k} S(c_{i_\ell}) = X$
- *there are k variables $x_{i_1}, x_{i_2}, \dots x_{i_k}$ such that $\forall \ell \in \{1, \dots, k\}$, $x_{i_\ell} \in S(c_{i_\ell})$ and $\bigcup_{1 \leq j \leq \ell} S(c_{i_j}) = \{x_i \mid i = 1, \dots, i_\ell\}$ hold.*

In words, the ordering must be such that the variables in the scope of one distinguished constraint (c_{i_1}) all appear first, then all the variables in the scope of some c_{i_2} (except for those already mentioned by c_{i_1}), etc. The variables $x_{i_1}, \dots, x_{i_k}$ in the definition are such that x_{i_j} is the last variable assigned in the scope of c_{i_j}. We refer to these variables as *milestones* in the ordering.

For instance, with the notation of the definition we must have $S(c_{i_1}) = \{x_1, x_2, \dots x_{i_1}\}$ and $S(c_{i_1}) \cup S(c_{i_2}) = \{x_1, x_2, \dots x_{i_1}, x_{i_1+1}, \dots x_{i_2}\}$. The variable x_{i_1} is a milestone (last variable assigned in the scope of c_{i_1}).

Under the assumption that such a variable ordering is used, we can give a generalization of Proposition 2.

Proposition 6 *Let P be an n-ary CSP, and assume that nBT explores the variables in some order compatible with the constraints in C. Then there is an injective mapping from the maximally deep consistent nodes (x_i, v_i) in the search tree such that x_i is a milestone, and the maximal cliques in $\mu_G(P)$.*

Proof: Let t be an assignment corresponding to a node as in the statement, and write x_{i_j} for the last variable assigned by t (which is a milestone by assumption). Similarly, let t' be another maximal consistent assignment with the milestone $x_{i_{j'}}$ as its last variable. Write T for the set $\{t[S(c)] \mid c \in C, S(c) \subseteq \{x_1, \dots, x_{i_j}\}\}$, that is, for the set of all projections of t onto the scopes of constraints fully assigned by t, and similarly for T'. Then T (resp. T') is included in some maximal clique $Cl(t)$ (resp. $Cl(t')$) of $\mu_G(P)$. Now assume $j' \geq j$ (wlog). From $t \neq t'$, the fact that t is maximally consistent, and the fact that t' is consistent, it follows that t' differs from t on at least one variable x_ℓ with $\ell \leq i_j$. Hence this variable is assigned differently by both, and there is some constraint c_{i_ℓ} such that $t[S(c_{i_\ell})]$ is different from $t'[S(c_{i_\ell})]$, and it follows that t, t' cannot be included in a common clique. Hence Cl defines an injective mapping from assignments as in the statement to maximal cliques in $\mu_G(P)$, as desired. □

Using this property, we can bound the number of nodes in a search tree induced by a backtracking search, and its time complexity, in terms of the generalized microstructure.

Proposition 7 *Let $P = (X, D, C)$ be an n-ary CSP, and assume that nBT uses a variable ordering which is compatible with the constraints in C. Then the number of nodes $N_{nBT}(P)$ in the search tree of nBT on P satisfies $N_{nBT}(P) \leq nd^a \cdot \omega_{\#}(\mu_G(P))$. Its time complexity is in $O(nea \cdot d^a \cdot \omega_{\#}(\mu_G(P)))$.*

Proof: From Proposition 6 it follows that the subtree induced by the search tree on milestones contains at most $\omega_{\#}(\mu_G(P))$ nodes. Now for reaching a milestone from the previous one, that is, for extending an assignment to $x_1, \ldots, x_{i_j}$ onto an assignment to $x_1, \ldots, x_{i_{j+1}}$ (with the notation of Definition 6), nBT explores at most a variables (this is by definition of an ordering compatible with the constraints). Hence it explores at most d^a combinations of values (nBT has no clue for ruling out an assignment before assigning all variables in the scope of a constraint). Since a branch contains at most n milestones, we get the result. The time complexity follows directly since each node requires at most e constraint checks, each one in time $O(a)$ with an appropriate data structure. □

A similar result holds for nFC$_i$ ($i \geq 2$). Nevertheless, we must consider the additional cost due to applying GAC, that is $O(e \cdot a \cdot r)$ at each node. However, note that contrary to nBT, due to the use of GAC, nFC$_i$ explores only the r tuples allowed by c when exploring the variables in $S(c)$, rather than all d^a combinations of values.

Proposition 8 *Let $P = (X, D, C)$ be an n-ary CSP, and assume that nFC$_i$ ($i \geq 2$) uses a variable ordering which is compatible with the constraints in C. Then the number of nodes $N_{nFC_i}(P)$ in the search tree of nFC$_i$ on P satisfies $N_{nFC_i}(P) \leq nr \cdot \omega_{\#}(\mu_G(P))$. Its time complexity is in $O(nea \cdot r^2 \cdot \omega_{\#}(\mu_G(P)))$.*

The same result also holds for nRFL.

As a final note, we have considered a total order in definition 6. A partial order can be used instead, provided the following holds: for each constraint c_{i_j} ($1 < j \leq k$), all the variables of $c_{i_{j-1}}$ have been assigned before the variables of c_{i_j} which do not belong to the scope of a previous constraint. The last assigned variable of each constraint will be the corresponding milestone variable. Moreover, the considered order may be dynamic.

4.3 Time Complexity without Ordering

Arguably, our restriction to variable orderings which are compatible with constraints is not met by all reasonable variable orderings. For instance, there is no reason in general for the well-known dom/deg heuristic to yield such orderings. However, we show here that such restrictions are necessary.

To show this, we build a family of instances which have a linear number of cliques in their generalized microstructure (in its number of vertices), but for which nFC$_5$ explores a search tree of size exponential (in the number of vertices) for some specific variable ordering.

The instances (X, D, C) in this family are built as follows. Writing e for the number of constraints, there are two distinguished variables x_0, x'_0 in X (common

to all constraints), and $X \setminus \{x_0, x'_0\}$ is partitioned into e sets $X_1, \ldots, X_e$ (X_i is specific to c_i). So each constraint $c_i \in C$ has scope $S(c_i) = \{x_0, x'_0\} \cup X_i$. Now D is $\{v_0, \ldots, v_{e-1}\}$ for all variables ($d = e$). Finally, the tuples allowed by the constraint c_i are precisely those of the form $\{(x_0, v_j), (x'_0, v_{i+j}), \ldots\}$, for $j = 1, \ldots, e$ (indices are taken modulo e) and unrestricted assignments to X_i. The point is that for $i \neq i'$, the restrictions of the tuples allowed by c_i and $c_{i'}$ onto $\{x_0, x'_0\}$ never match, so that there are no edges in $\mu_G(P)$. Hence the number of cliques in $\mu_G(P)$ is exactly its number of vertices $|V| = e^{2+(n-2)/e}$.

On the other hand, assume that nFC$_5$ explores all variables in the X_i's and only explores x_0, x'_0 after them. Then because all values for x_0 have a support in all constraints, and similarly for x'_0, no value will be removed before reaching x_0 or x'_0, and hence all $e^{n-2} \sim |V|^e$ combinations of values will be explored, that is, exponentially more than the number of cliques in $\mu_G(P)$.

In the general case, we can bound the time complexity of nFC$_i$ as follows.

Proposition 9 *Let $P = (X, D, C)$ be an n-ary CSP, and assume that nFC$_i$ ($i \geq 2$) uses a variable ordering such that the maximum number of non-milestone variables assigned consecutively is m. Then the number of nodes $N_{nFC_i}(P)$ in the search tree of nFC$_i$ on P satisfies $N_{nFC_i}(P) \leq nd^m \cdot \omega_{\#}(\mu_G(P))$. Its time complexity is in $O(nea \cdot rd^m \cdot \omega_{\#}(\mu_G(P)))$.*

In our previous example, we have $m = n - 2$.

5 A Few Tractable Classes for Backtracking

The number of cliques in a graph can grow exponentially with the size of the graph [20], and so can the number $\omega_{\#}(G)$ of *maximal* cliques in a graph G [21]. However, for some classes of graphs, the number of maximal cliques can be bounded by a polynomial in the size of the graph. If the (generalized) microstructure of a (family of) CSP P belongs to one of these classes, then our analysis in the previous sections allows us to conclude that P is solved in polynomial time by classical backtracking algorithms, *without the need to recognize the instance to be in the class*. In this section, we study several such classes of graphs in terms of their relevance to constraint satisfaction problems.

5.1 Triangle-Free and Bipartite Graphs

Recall that a *k-cycle* in a graph $G = (V, E)$ is a sequence $(v_1, v_2, \ldots v_{k+1})$ of distinct vertices satisfying $\forall i, 1 \leq i \leq k, \{v_i, v_{i+1}\} \in E$, and $v_1 = v_{k+1}$.

A *triangle-free* graph is an undirected graph with no 3-cycle. It is easily seen that the number of maximal cliques in a triangle-free graph is exactly its number of edges E. Hence by our analysis, if a class of CSPs has a triangle-free (generalized) microstructure, algorithms (n)BT, (n)FC and (n)RFL correctly solve them in polynomial time. Note however that this is quite a degenerate case, since except for instances having at most two variables (binary case) or two constraints (non-binary case), instances with a triangle-free (generalized) microstructure are inconsistent.

Another degenerate but illustrative case is one of bipartite graphs. A graph is *bipartite* if it does not contain an odd cycle. Again, a bipartite graph cannot contain any clique of more than two vertices, and hence no partial assignment to more than three variables will ever be considered by BT (hence it obviously runs in time $O(d^3)$).

We now turn to more interesting classes, which also essentially contain inconsistent CSPs, but for which our analysis gives a better time complexity than the classical one.

5.2 Planar, Toroidal, and all Embedded Graphs

Definition 7 (planar) *A* planar *graph is a graph that can be embedded in the plane without crossing edges.*

[20] proved that the number of cliques in a planar graph is at most $8(|V|-2)$.

Definition 8 (toroidal) *A graph is* toroidal *if it can be embedded on the torus without crossing edges.*

[22] showed that every toroidal graph has at most $8(|V|+9)$ cliques and then that every graph embeddable in some surface has a linear number of cliques ($8(|V|+27)$ at worst). Since the microstructure $\mu(P)$ (resp. $\mu_G(P)$) of a CSP P contains at most nd vertices (resp. er vertices), it follows that if $\mu(P)$ (resp. $\mu_G(P)$) belongs to one of these classes of graphs, then $\omega_\#(\mu(P))$ (resp. $\omega_\#(\mu_G(P))$) is in $O(nd)$ (resp. $O(er)$). Thanks to Prop. 3, 4, 7 and 8, we immediately get the following.

Theorem 1 *Let Em denote the class of all CSPs whose microstructure is planar, toroidal, or embeddable in a surface. Then instances in Em are solved in time*

- $O(n^2d \cdot \omega_\#(\mu(P))) = O(n^3d^2)$ *by BT or FC,*
- $O(ned^2 \cdot \omega_\#(\mu(P))) = O(n^2ed^3)$ *by RFL,*
- $O(nead^a \cdot \omega_\#(\mu_G(P))) = O(ne^2ard^a)$ *by nBT,*
- $O(near^2 \cdot \omega_\#(\mu_G(P))) = O(ne^2ar^3)$ *by nFC$_i$, nRFL.*

Recall that this family of graphs cannot contain as a minor, an 8-clique (for toroidal graphs), or a 5-clique nor $K_{3,3}$ (for planar graphs). It follows in particular that any binary (resp. non-binary) CSP in *Em* over at least 8 variables (resp. constraints) is inconsistent. Hence again this class is a little degenerate, however a classical analysis states that, e.g., BT solves these instances in time $O(d^8)$. In case d is large, this is looser that $O(n^3d^2)$.

5.3 CSG Graphs

We finally turn to the class of *CSG graphs*, which has been introduced by [23] and which generalizes the class of chordal graphs. Given a graph (V, E) and an ordering $v_1, \ldots, v_{|V|}$ of its vertices, we write $N^+(v_i)$ for the *forward neighborhood* of v_i, that is, $N^+(v_i) = \{v_j \in V | \{v_i, v_j\} \in E, i < j\}$. For $V' \subseteq V$, we write $G(V')$ for the graph *induced* by E on V', namely, $G(V') = (V', E')$ where $E' = \{\{x, y\} \mid x, y \in V' \text{ and } \{x, y\} \in E\}$.

Definition 9 (CSG graphs) *The class of graphs CSG^k is defined recursively as follows.*

- *CSG^0 is the class of complete graphs.*
- *Given $k > 0$, CSG^k is the class of graphs $G = (V, E)$ such that there exists an ordering $\sigma = (v_1, ..., v_{|V|})$ of V satisfying that for $i = 1, \ldots, |V|$, the graph $G(N^+(v_i))$ is a CSG^{k-1} graph.*

The class of CSG graphs generalizes the class of complete graphs (CSG^0 graphs) and the class of chordal graphs (CSG^1 graphs). Like chordal graphs, CSG graphs have nice properties. For instance, they can be recognized in polynomial time. Moreover, Chmeiss and Jégou have proved that CSG^k graphs have at most $|V|^k$ maximal cliques, and they have proposed an algorithm running in time $O(|V|^{2(k-1)}(|V| + |E|))$ for finding all of them.

These two algorithms qualify the class of all CSPs which have a CSG^k (generalized) microstructure as a tractable class for any fixed k. We are however able to show that even a *generic* algorithm such as (n)BT, FC, nFC_i or (n)RFL runs in polynomial time on such CSPs, without even the need to recognize membership in this class, nor to compute the microstructure. Again, the result follows from the number of maximal cliques together with Prop. 3, 4, 7 and 8.

Theorem 2 *Given any integer k, the class of all CSPs which have a CSG^k (generalized) microstructure is solved in time*

- *$O(n^2 d \cdot \omega_{\#}(\mu(P))) = O(n^{k+2} d^{k+1})$ by BT and FC,*
- *$O(ned^2 \cdot \omega_{\#}(\mu(P))) = O(n^{k+1} e d^{k+2})$ by RFL,*
- *$O(nead^a \cdot \omega_{\#}(\mu_G(P))) = O(ne^{k+1} a r^k d^a)$ by nBT,*
- *$O(near^2 \cdot \omega_{\#}(\mu_G(P))) = O(ne^{k+1} a r^{k+2})$ by nFC_i or nRFL.*

Observe that even the time complexities are better than those of the dedicated algorithm. For instance, the latter computes the microstructure of a binary CSP and enumerates all maximal cliques until exhaustion, or an n-clique is found. So it has a time complexity in $O((nd)^{2(k-1)}(nd + n^2 d^2)) = O((nd)^{2k})$. Nevertheless, we can note that this algorithm is defined for general CSG^k graphs while (generalized) microstructures of CSP are very particular graphs.

CSG graphs are generally speaking less restrictive than the previous classes of graphs. For instance, it is possible to have CSG graphs with n-cliques (resp. e-clique) for any value of n (resp. e), contrary to the case of planar graphs. In particular, there are consistent CSPs with a CSG^k (generalized) microstructure. It is the case for CSG^0 which are consistent binary CSPs with monovalent domains (one value per domain) or consistent non-binary CSPs with exactly one allowed tuple per relation. Nevertheless, CSPs which have a CSG^1 (generalized) microstructure can be consistent or not and it is easy to build a CSP with several solutions, which corresponds to a collection of cliques of size n (binary case) or e (non-binary case). Furthermore, unlike previous classes, in CSG^k graphs (with $k \geq 1$), there is no restrictions on the values of n, d, e, a or r.

For instance, the *Hanoi* or *Domino* benchmarks used in the CSP solver competitions (e.g. [24]) have a CSG^1 microstructure after applying an AC filtering.

Thus, the use of algorithms such as RFL allows them to exploit the tractable classes that we have highlighted here. However, this set of classes of CSPs still has to be studied in detail for assessing its practical interest.

6 Discussion and Perspectives

We have investigated the time complexity of classical, generic algorithms for solving CSPs under a new perspective. Our analysis expresses the complexity in terms of the number of maximal cliques in the (generalized) microstructure of the CSP to be solved. Our analysis reveals that essentially, backtracking and forward checking visit each maximal clique in the (generalized) microstructure at most once. From this analysis we derived tractable classes of CSPs, which can be solved by classical algorithms in polynomial time, *without the need to recognize that the instance at hand is in the class.* So, the results obtained shed a new light on the analysis of CSPs.

Some relationships between tractable classes presented here and other tractable classes of the state of the art are simple to identify. For example, for the classes defined by the structure, that is, CSPs of bounded width, it is easy to see that they are incomparable. It is possible to build a CSP whose constraint network is a complete graph with a polynomial number of maximal cliques. E.g. consider a CSPs defined on n variables with the same domain of size d and with equality constraints between every pair of variables. For this kind of instances, there are d maximal cliques of size n corresponding to the d solutions of the CSP. Conversely, it is easy to see that an acyclic binary CSP can have an exponential number of solutions, and thus, an exponential number of maximal cliques.

For hybrid classes, we can see that the tractable class of binary CSPs satisfying the Broken Triangle Property (BTP) [25] is also incomparable. It is very simple to define a CSP satisfying the BTP property with an exponential number of solutions and thus an exponential number of maximal cliques. In contrast, one can easily define a binary CSP whose microstructure is planar and which does not satisfy the BTP property. For instance, the CSP defined on three variables with domains $D(x_1) = \{a, b\}, D(x_2) = \{c, d\}, D(x_3) = \{e, f\}$, and pairs of values $(a, d), (b, e), (c, f)$ prohibited.

It is thus clear that the tractable classes which we propose here are not necessarily weaker or stronger than the classes of the state of the art. Their main advantage is that they can simply be treated without recourse to ad hoc algorithms, but with state-of-the-art solvers. Despite the lack of a theoretical difficulty in the definition of these tractable classes, they seem to offer an advantage compared to many tractable classes which are for most of them, artificial, if one refers to real benchmarks.

The first perspective of this work is to investigate more classes of graphs with polynomially many maximal cliques. Of particular interest here is the study by [26], who precisely characterize these classes of graphs in terms of intersection graphs. Another important perspective is to relate our analysis to other tractable classes obtained in different manners, in the spirit of the first comparisons given above. An important perspective is also to extend our study to the other possible

generalizations of the microstructure for non-binary constraint satisfaction problems. In [27], a close work is presented since it proposes to define a new tractable class which is based on the polynomial size of search trees. The relations with our work should be now clarified.

To complete our study, we must also address the conjecture 1 for the MAC algorithm. Actually, we strongly believe that this conjecture is true, but we have not been able to prove this fact, especially due to the existence of the concept of negative decisions for MAC.

Finally, it would be interesting to analyse a wide set of benchmarks - particularly the ones easily solved by solvers of the state of the art - to investigate cases in which the tractable classes exhibited here are present, in the spirit of the first observations made on benchmarks such as *Hanoi* or *Domino*.

Acknowledgments

Philippe Jégou would like to thank Maria Chudnovsky for their fruitful discussion, about graph theory, perfect graphs and links with classes of graphs related to the clique problem, and the links with the microstructure of CSPs.

References

1. F. Rossi, P. van Beek, and T. Walsh. *Handbook of Constraint Programming*. Elsevier, 2006.
2. R. Haralick and G. Elliot. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–313, 1980.
3. B. Nadel. *Tree Search and Arc Consistency in Constraint-Satisfaction Algorithms*, pages 287–342. In *Search in Artificial Intelligence*. Springer-Verlag, 1988.
4. D. Sabin and E. Freuder. Contradicting Conventional Wisdom in Constraint Satisfaction. In *Proc. of ECAI*, pages 125–129, 1994.
5. C. Bessière, P. Meseguer, E. C. Freuder, and J. Larrosa. On forward checking for non-binary constraint satisfaction. *Artificial Intelligence*, 141:205–224, 2002.
6. E. Freuder. A Sufficient Condition for Backtrack-Free Search. *JACM*, 29 (1):24–32, 1982.
7. G. Gottlob, N. Leone, and F. Scarcello. A Comparison of Structural CSP Decomposition Methods. *Artificial Intelligence*, 124:343–282, 2000.
8. S. de Givry, T. Schiex, and G. Verfaillie. Dcomposition arborescente et cohérence locale souple dans les CSP pondérés. In *Proceedings of JFPC'06*, 2006.
9. Lisa Purvis and Peter Jeavons. Constraint tractability theory and its application to the product development process for a constraint-based scheduler. In *Proceedings of the 1st International Conference on The Practical Application of Constraint Technologies and Logic Programming*, pages 63–79, 1999. This paper was awarded First Prize in the Constraints Technologies area of PACLP'99.
10. Antoine Rauzy. Polynomial restrictions of SAT: What can be done with an efficient implementation of the Davis and Putnam's procedure. In U. Montanari and F. Rossi, editors, *Proc. International Conference on Principles of Constraint Programming (CP 1995)*, pages 515–532. Springer Verlag, 1995.

11. Philippe Jégou, Samba Ndiaye, and Cyril Terrioux. A new evaluation of forward checking and its consequences on efficiency of tools for decomposition of csps. In *ICTAI (1)*, pages 486–490, 2008.
12. Sebastian Ordyniak, Daniël Paulusma, and Stefan Szeider. Satisfiability of acyclic and almost acyclic cnf formulas. In *FSTTCS*, pages 84–95, 2010.
13. P. Jégou. Decomposition of Domains Based on the Micro-Structure of Finite Constraint Satisfaction Problems. In *Proceedings of AAAI 93*, pages 731–736, Washingtown, DC, 1993.
14. M. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.
15. David A. Cohen. A New Classs of Binary CSPs for which Arc-Constistency Is a Decision Procedure. In *Proceedings of CP 2003*, pages 807–811, 2003.
16. András Salamon and Peter Jeavons. Perfect Constraints Are Tractable. In *Proceedings of CP*, pages 524–528, 2008.
17. Achref El Mouelhi. Generalized micro-structures for non-binary csp. In *Doctoral Programme CP, http://zivny.cz/dp12/*, pages 13–18, 2012.
18. R. Dechter and J. Pearl. Tree-Clustering for Constraint Networks. *Artificial Intelligence*, 38:353–366, 1989.
19. F. Rossi, C. Petrie, and V. Dhar. On the equivalence of constraint satisfaction problems. In *Proceedings of the 9th European Conference on Artificial Intelligence*, pages 550–556, 1990.
20. David R. Wood. On the maximum number of cliques in a graph. *Graphs and Combinatorics*, 23:337–352, June 2007.
21. J. W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3:23–28, 1965.
22. Vida Dujmovic, Gasper Fijavz, Gwenaël Joret, Thom Sulanke, and David R. Wood. On the maximum number of cliques in a graph embedded in a surface. *European J. Combinatorics*, 32(8):1244–1252, 2011.
23. A. Chmeiss and P. Jégou. A generalization of chordal graphs and the maximum clique problem. *Information Processing Letters*, 62:111–120, 1997.
24. Third International CSP Solver Competition , 2008. http://cpai.ucc.ie/08.
25. M. Cooper, Peter Jeavons, and Andras Salamon. Generalizing constraint satisfaction on trees: hybrid tractability and variable elimination. *Artificial Intelligence*, 174:570–584, 2010.
26. Bill Rosgen and Lorna Stewart. Complexity results on graphs with few cliques. *Discrete Mathematics and Theoretical Computer Science*, 9:127–136, 2007.
27. David A. Cohen, Martin C. Cooper, Martin J. Green, and Dániel Marx. On guaranteeing polynomially bounded search tree size. In *CP*, pages 160–171, 2011.